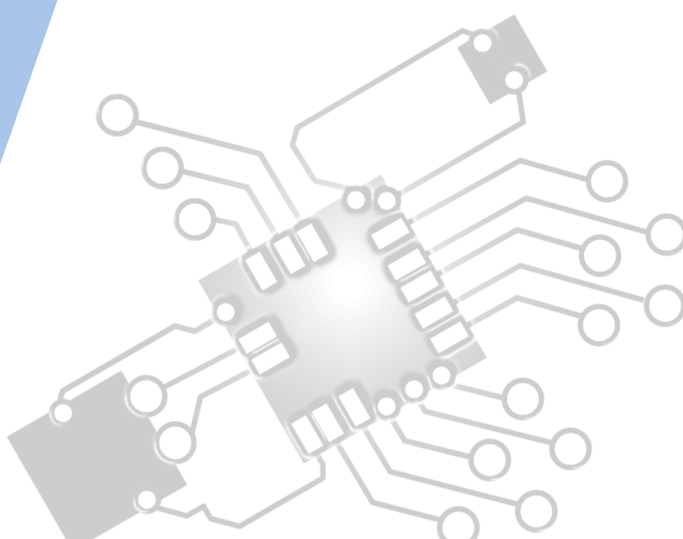
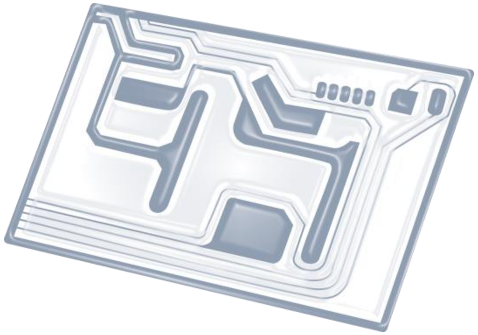
Planning & system installation

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL only 5 Overview

Thinking recursively

5.1.1 Identify a situation that requires the use of recursive thinking
5.1.2 Identify recursive thinking in a specified problem solution
5.1.3 Trace a recursive algorithm to express a solution to a problem

Abstract data structures

5.1.4 Describe the characteristics of a two-dimensional array
5.1.5 Construct algorithms using two-dimensional arrays
5.1.6 Describe the characteristics and applications of a stack
5.1.7 Construct algorithms using the access methods of a stack
5.1.8 Describe the characteristics and applications of a queue
5.1.9 Construct algorithms using the access methods of a queue
5.1.10 Explain the use of arrays as static stacks and queues

Linked lists

5.1.11 Describe the features and characteristics of a dynamic data structure
5.1.12 Describe how linked lists operate logically
5.1.13 Sketch linked lists (single, double and circular)

Trees

5.1.14 Describe how trees operate logically (both binary and non-binary)
5.1.15 Define the terms: parent, left-child, right-child, subtree, root and leaf
5.1.16 State the result of inorder, postorder and preorder tree traversal
5.1.17 Sketch binary trees

Applications

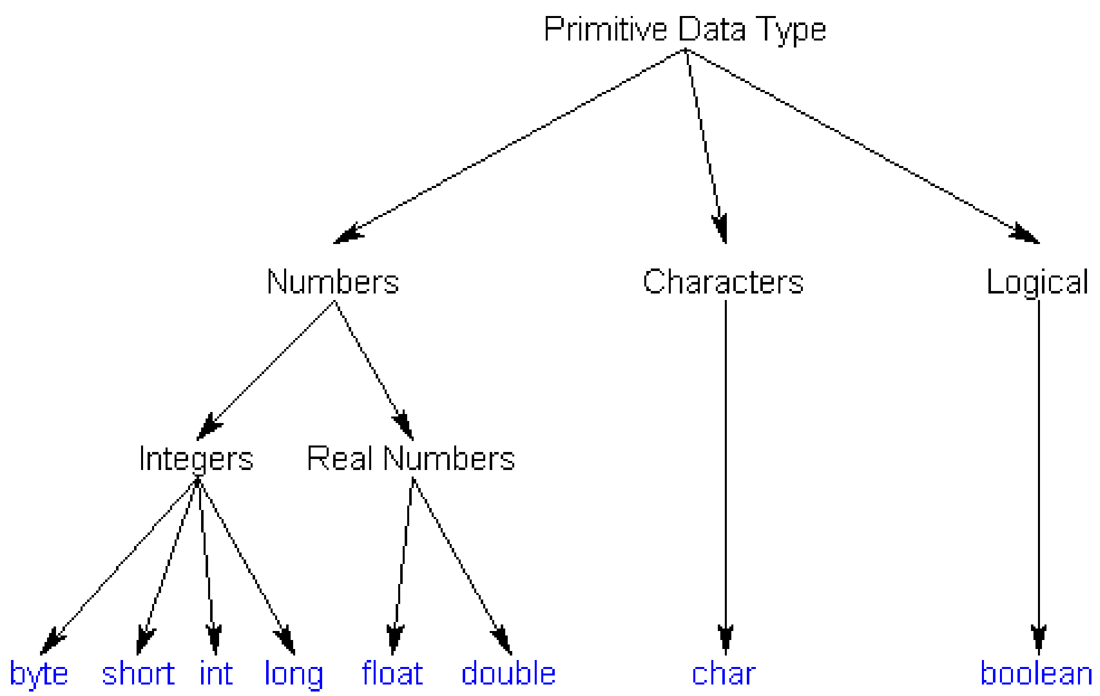
5.1.18 Define the term dynamic data structure
5.1.19 Compare the use of static and dynamic data structures
5.1.20 Suggest a suitable structure for a given situation

Topic 5.1.6

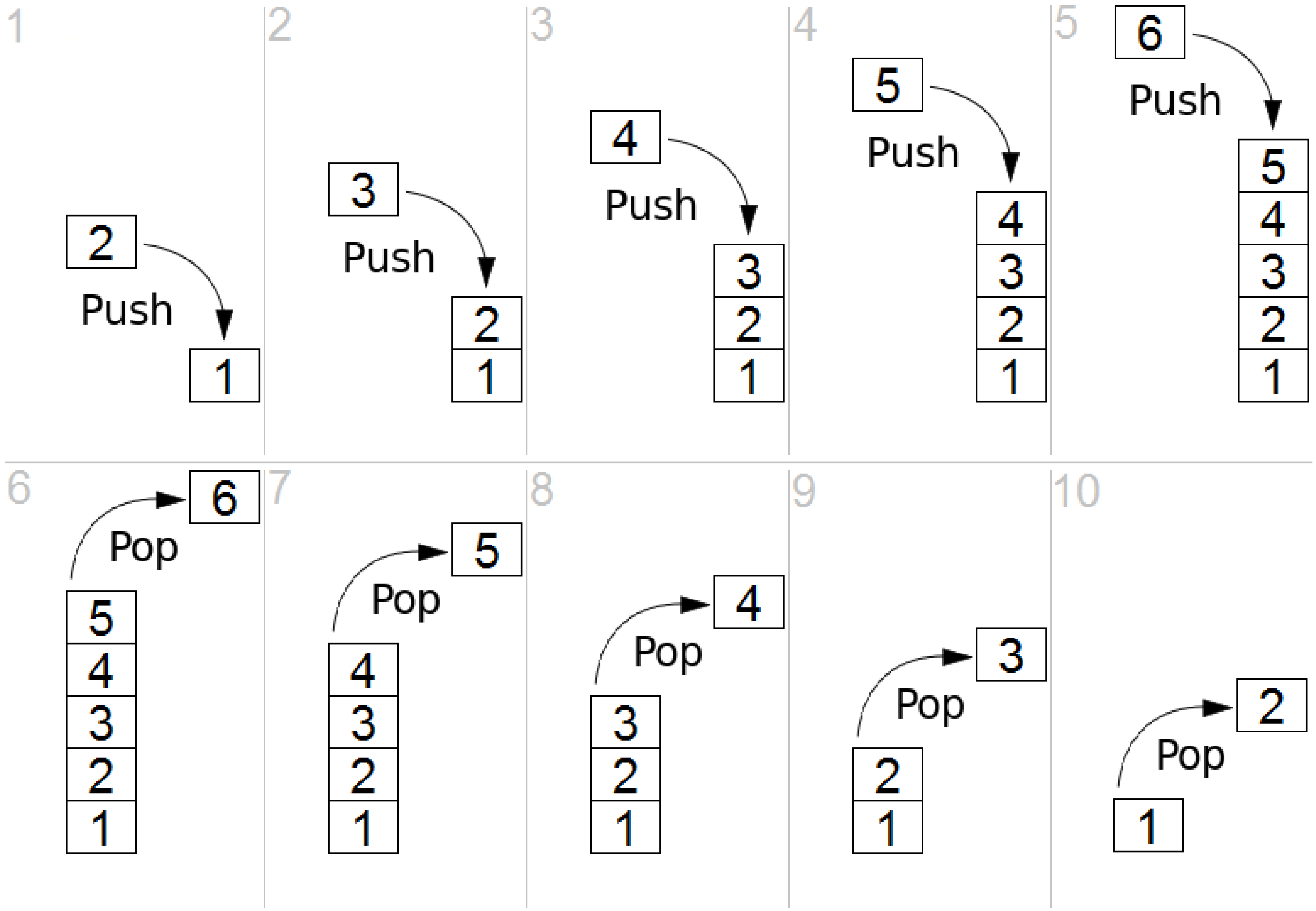
Describe the **characteristics** and **applications** of a **stack**

Abstract Data Structures (ADTs)

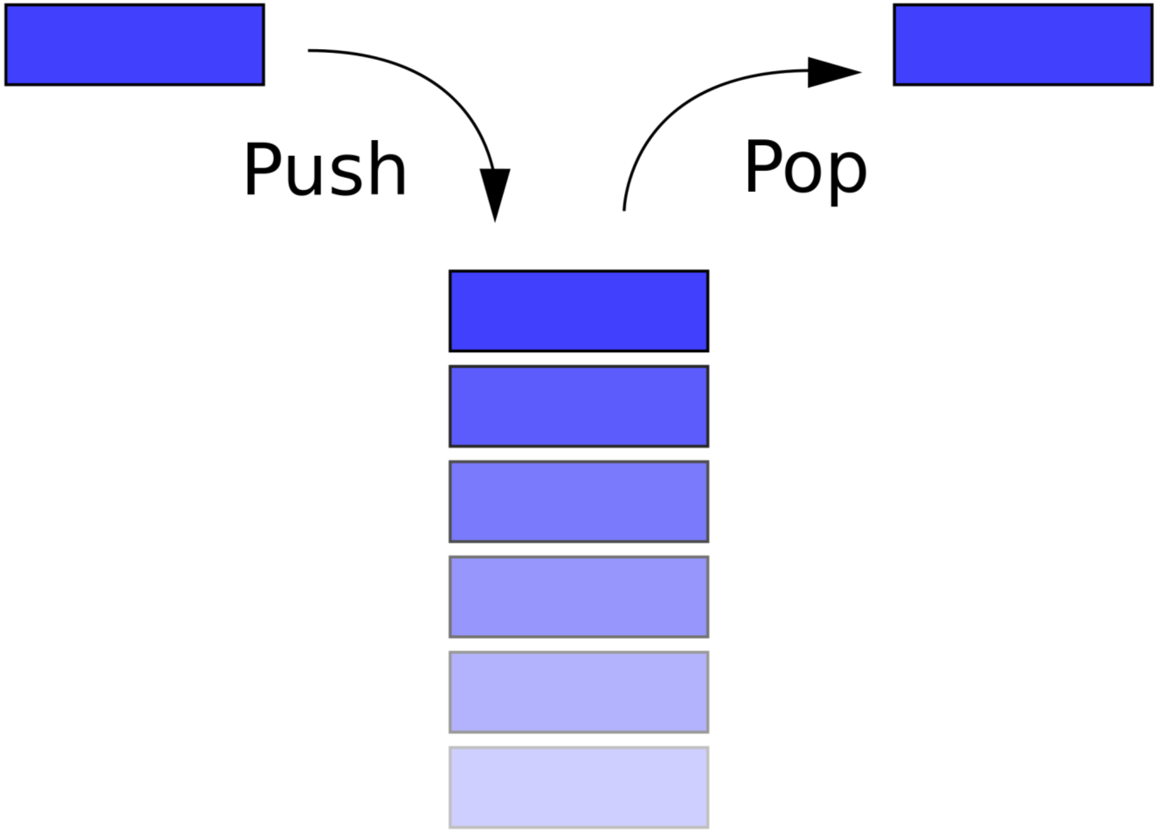
- 2D array
- **Stack**
- Queue
- Linked List
- (Binary) Tree



Stacks – all about the PUSH and POP

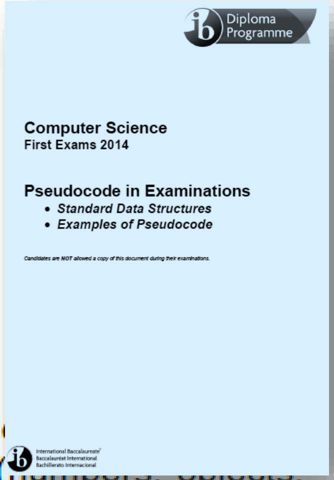


Last in, First out



LIFO

3 Stack Methods



Stacks

A stack stores a set of elements in a particular order: Items are retrieved in th[e reverse order in] which they are inserted (Last-in, First-out). The elements may be of any type (numbers, objects, arrays, Strings, etc.).

Method name	Brief description	Example: OPS, a stack of integers	Comment
`push()`	Push an item onto the stack	`OPS.push(42)`	Adds an element that contains the argument, whether it is a value, String, object, etc. to the top of the stack.
`pop()`	Pop an item off the stack	`NUM = OPS.pop()`	Removes and returns the item on the top of the stack.
`isEmpty()`	Test: stack contains no elements	`if OPS.isEmpty() then`	Returns TRUE if the stack does not contain any elements.

Example 1: Move from array to stack

Write an algorithm that will move all the elements from a linear integer array LINE to a stack called S.

```
int COUNTER = 0
loop COUNTER from 0 to LINE.length
    S.push(LINE[COUNTER])
end loop
```

Example 2: Print out a stack

Write an algorithm that will print all the String values of a stack called S.

```
loop while not S.isEmpty()
    output( S.pop() )
end loop
```

Example

```java
import java.util.*;

public class StackDemo {

   static void showpush(Stack st, int a) {
      st.push(new Integer(a));
      System.out.println("push(" + a + ")");
      System.out.println("stack: " + st);
   }

   static void showpop(Stack st) {
      System.out.print("pop -> ");
      Integer a = (Integer) st.pop();
      System.out.println(a);
      System.out.println("stack: " + st);
   }

   public static void main(String args[]) {
      Stack st = new Stack();
      System.out.println("stack: " + st);
      showpush(st, 42);
      showpush(st, 66);
      showpush(st, 99);
      showpop(st);
      showpop(st);
      showpop(st);
      try {
         showpop(st);
      } catch (EmptyStackException e) {
         System.out.println("empty stack");
      }
```

```
stack: [ ]
push(42)
stack: [42]
push(66)
stack: [42, 66]
push(99)
stack: [42, 66, 99]
pop -> 99
stack: [42, 66]
pop -> 66
stack: [42]
pop -> 42
stack: [ ]
pop -> empty stack
```

Ex 2

```
public class StackExample {

    public static void main(String[] args) {

        Stack<String> sk=new Stack<String>();

        sk.push("a");
        sk.push("c");
        sk.push("e");
        sk.push("d");

        Iterator it=sk.iterator();

        System.out.println("Size before pop() :"+sk.size());

        while(it.hasNext())
        {
            String iValue=(String)it.next();
            System.out.println("Iterator value :"+iValue);
        }

        // get and remove last element from stack
        String value =(String)sk.pop();

        System.out.println("value :"+value);

        System.out.println("Size After pop() :"+sk.size());
```